



Zachodniopomorski Uniwersytet Technologiczny
w Szczecinie

Wydział Inżynierii Mechanicznej i Mechatroniki

Projekt z przedmiotu
Inżynieria Oprogramowania

Sprawozdanie

**Tytuł projektu: Aplikacja napisana w języku C# służąca
do translacji (i dokonywania przeglądu) tekstu w
formie cyrylicy na tekst alfabetu polskiego z
zachowaniem zgodności fonetycznej.**

Kierunek/semestr:

Opracowali i zaprojektowali:

Data oddania Projektu:

Mechatronika/VI

Abelite Tomasz, Kiłyk Julian, Soczalski Mateusz

15.06.2021 r.

Spis treści

1. Wstęp teoretyczny	2
1.1. Cyrylica – czym jest?	2
1.2. Współczesna cyrylica rosyjska	2
2. Opis aplikacji.....	4
2.1. Środowisko Visual Studio.....	4
2.2. Krótki opis aplikacji.....	4
2.3. Back-end aplikacji	5
2.4. Front-end.....	6
3. Wnioski	10

1. Wstęp teoretyczny

1.1. Cyrylica – czym jest?

Cyrylica – pismo alfabetyczne służące do zapisu języków wschodniosłowiańskich, większości południowosłowiańskich i innych. Nazwa nawiązuje do apostoła Słowian – św. Cyryla, który wspólnie ze św. Metodym, prowadząc misję wielkomorawską wśród Słowian, zapisał i wprowadził do liturgii język słowiański. Do zapisu tego języka zostały stworzone dwa alfabety – najpierw głagolica, z której zostały później zapożyczone niektóre litery cyrylicy, a potem cyrylica.

Za twórcę cyrylicy uznaje się któregoś z uczniów **Cyryla i Metodego**, przy czym najczęściej przywołuje się tu imiona Klimenta Ochrydzkiego bądź Konstantyna Presławskiego. Pierwsze ślady użycia cyrylicy pochodzą z terenów wschodniej Bułgarii. Za najstarszy datowany zabytek cyrylicy uważana jest dobrudżańska inskrypcja cara Piotra, datowana na 943 rok. W ciągu X–XII w. cyrylica rozprzestrzeniła się z Bułgarii na tereny Serbii oraz Rusi Kijowskiej.

1.2. Współczesna cyrylica rosyjska

Tabela poniższa przedstawia zgrupowane znaki aktualnie należące do cyrylicy rosyjskiej.

Litera	Kursywa	Transliteracja ^[3]	Transkrypcja ^[4]		
			Na początku wyrazu	W środku/na końcu wyrazu	
				Po samogłosce	Po spółgłosce
А а	<i>А а</i>	A a	a		
Б б	<i>Б б</i>	B b	b		
В в	<i>В в</i>	V v	w		
Г г	<i>Г г</i>	G g	g		
Д д	<i>Д д</i>	D d	d		
Е е	<i>Е е</i>	E e	je		ie, e
Ё ё	<i>Ё ё</i>	Ė ė	jo		io, o
Ж ж	<i>Ж ж</i>	Ž ž	ž		
З з	<i>З з</i>	Z z	z		
И и	<i>И и</i>	I i	i	ji	i, y
Й й	<i>Й й</i>	J j	j		
К к	<i>К к</i>	K k	k		
Л л	<i>Л л</i>	L l	ł, l		
М м	<i>М м</i>	M m	m		
Н н	<i>Н н</i>	N n	n		
О о	<i>О о</i>	O o	o		
П п	<i>П п</i>	P p	p		
Р р	<i>Р р</i>	R r	r		
С с	<i>С с</i>	S s	s		
Т т	<i>Т т</i>	T t	t		
У у	<i>У у</i>	U u	u		
Ф ф	<i>Ф ф</i>	F f	f		
Х х	<i>Х х</i>	H h	ch		
Ц ц	<i>Ц ц</i>	C c	c		
Ч ч	<i>Ч ч</i>	Č č	cz		
Ш ш	<i>Ш ш</i>	Š š	sz		
Щ щ	<i>Щ щ</i>	Š š	szcz		
Ъ ъ	<i>Ъ ъ</i>	'	nieużywany	jer (znak) twardy	
Ы ы	<i>Ы ы</i>	Y y	y		
Ь ь	<i>Ь ь</i>	'	nieużywany	' jer (znak) miękki lub pomija się ¹⁾	
Э э	<i>Э э</i>	È è	e		
Ю ю	<i>Ю ю</i>	Ů ů	ju		iu, u
Я я	<i>Я я</i>	Â â	ja		ia, a

Rysunek 1.1. – Tabela zawierająca znaki należące do cyrylicy rosyjskiej

[źródło: pl.wikipedia.org/wiki/Cyrylica]

1.3. Front-end i Back-end

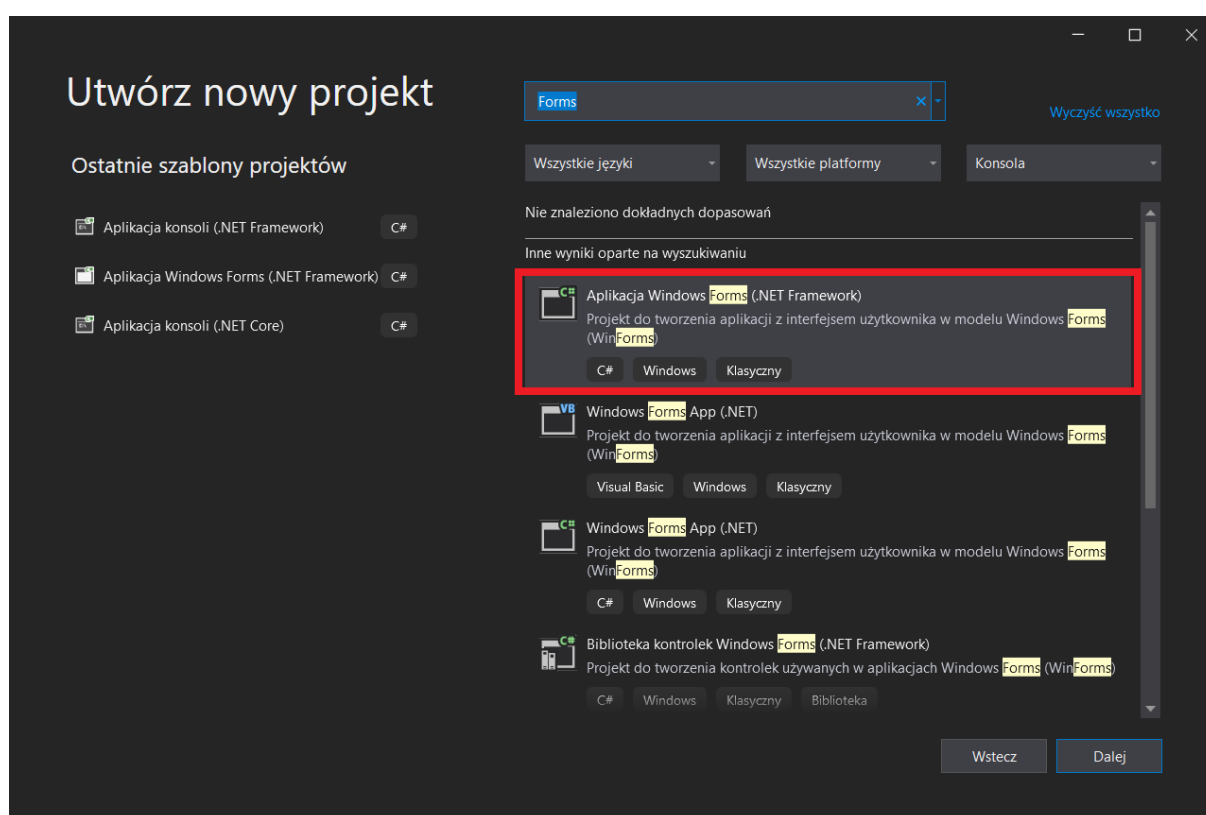
Front-end jest odpowiedzialny za pobieranie danych od użytkownika oraz przekazanie ich do **back-endu**. Następnie **back-end** na podstawie tych danych wykonuje określone zadanie. Opcjonalnie **front-end** może pokazać użytkownikowi wyniki otrzymane od **back-endu**. Często stosowanym tłumaczeniem jest „**fasada**” i „**wnętrze**”.

Terminy front-end i back-end są najczęściej stosowane w tej dziedzinie i zazwyczaj odnoszą się do nakładek zapewniających graficzny lub tekstowy interfejs (front-end) dla aplikacji konsolowych (back-end).

2. Opis aplikacji

2.1. Środowisko Visual Studio

Aplikacja została stworzona w środowisku Visual Studio w języku C# jako **Windows Forms (.NET Framework)**. Jest to typ projektu służący do tworzenia aplikacji z interfejsem użytkownika w modelu Windows Forms (WinForms).



Rysunek 2.1. – Menu wyboru projektu w Visual Studio

Visual Studio to zintegrowane środowisko programistyczne firmy Microsoft. Jest używane do tworzenia oprogramowania konsolowego oraz z graficznym interfejsem użytkownika, w tym aplikacji Windows Forms, WPF, Web Sites, Web Applications i inne. Aplikacje mogą być pisane na platformy: Microsoft Windows, Windows Phone, Windows CE, .NET Framework, Microsoft Silverlight, Linux, MacOS oraz konsole XBOX.

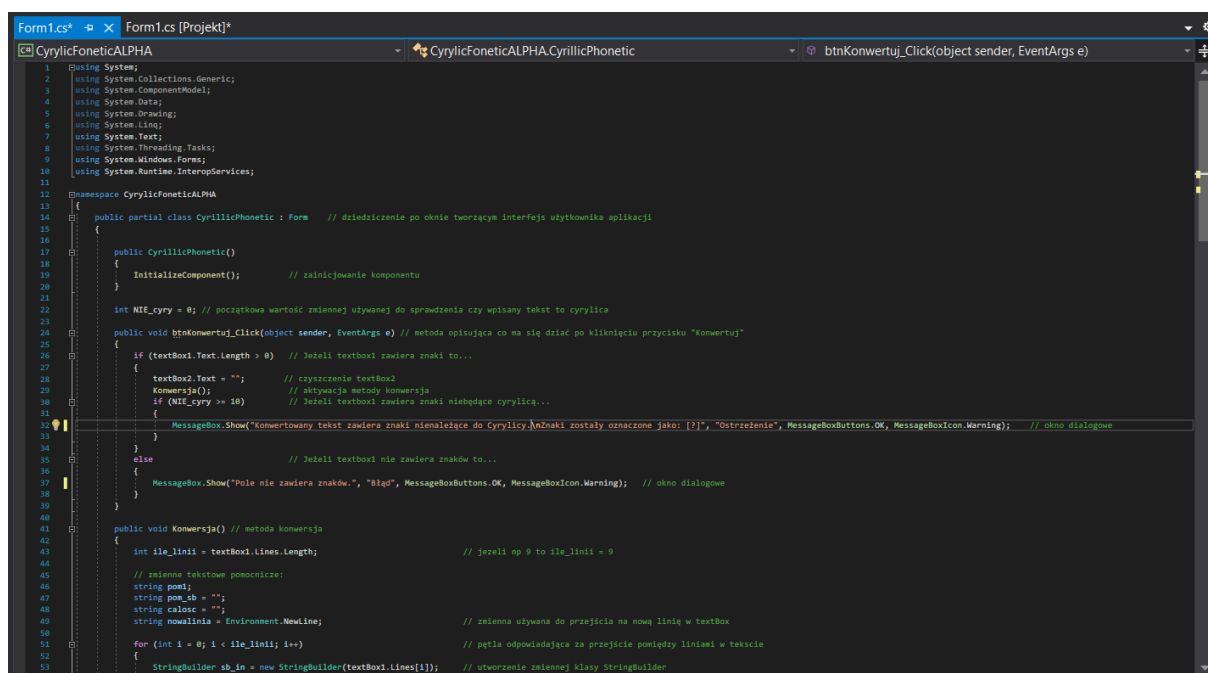
2.2. Krótki opis aplikacji

Z racji dużej rozbieżności w własnościach alfabetu polskiego oraz cyrylicy rosyjskiej czytanie znaków i powstałych z nich wyrazów może wydać się niezwykle skomplikowane

dla osoby nie mającej uprzednio styczności z językami wschodnio- lub południowosłowiańskimi. **Narzędzie jakim jest aplikacja CyrillicPhonetic 1.0.3** ma przyspieszyć zapoznanie z fonetyczną częścią tekstu pisanego w cyrylicy użytkownikowi operującemu alfabetem polskim. Program posłużyć może jako pomoc przy prowadzeniu konwersacji, w nauce lub przy czytaniu czasopism lub tekstów utworów muzycznych zapisanych cyrylicy.

2.3. Back-end aplikacji

Poniższy kod został zawarty w skrypcie Visual studio:



```
1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel;
4 using System.Data;
5 using System.Drawing;
6 using System.Linq;
7 using System.Text;
8 using System.Threading.Tasks;
9 using System.Windows.Forms;
10 using System.Runtime.InteropServices;
11
12 namespace CyrillicPhoneticALPHA
13 {
14     public partial class CyrillicPhonetic : Form // dziedziczenie po oknie tworzącym interfejs użytkownika aplikacji
15     {
16         public CyrillicPhonetic()
17         {
18             InitializeComponent(); // zainicjowanie komponentu
19         }
20
21         int MIE_cyry = 0; // początkowa wartość zmiennej używanej do sprawdzenia czy wpisany tekst to cyrylica
22
23         public void btnKonwertuj_Click(object sender, EventArgs e) // metoda opisująca co ma się dziać po kliknięciu przycisku "konwertuj"
24         {
25             if (textBox1.Text.Length > 0) // jeżeli textbox zawiera znaki to...
26             {
27                 textBox2.Text = ""; // czyszczenie textbox2
28                 Konwersja(); // aktywacja metody konwersja
29                 if (MIE_cyry >= 10) // jeżeli textbox zawiera znaki niebędące cyrylicą...
30                 {
31                     MessageBox.Show("Konwertowany tekst zawiera znaki nie należące do Cyrylicy. Znaki zostały oznaczone jako: [?]", "Ostrzeżenie", MessageBoxButtons.OK, MessageBoxIcon.Warning); // okno dialogowe
32                 }
33             }
34             else // jeżeli textbox nie zawiera znaków to...
35             {
36                 MessageBox.Show("Pole nie zawiera znaków.", "Błąd", MessageBoxButtons.OK, MessageBoxIcon.Warning); // okno dialogowe
37             }
38         }
39
40         public void Konwersja() // metoda konwersja
41         {
42             int ile_linii = textBox1.Lines.Length; // jeżeli np 9 to ile_linii = 9
43
44             // zmienne tekstowe pomocnicze:
45             string pom1;
46             string nowa_sb = "";
47             string calosc = "";
48             string nowalini = Environment.NewLine; // zmienna używana do przejścia na nową linię w textbox
49
50             for (int i = 0; i < ile_linii; i++) // pętla odpowiadająca za przejście pomiędzy liniami w tekście
51             {
52                 StringBuilder sb_in = new StringBuilder(textBox1.Lines[i]); // utworzenie zmiennej klasy StringBuilder
53             }
54         }
55     }
56 }
```

Rysunek 2.2. – Fragment 1 skryptu z Visual Studio

```

Form1.cs* [Projekt]*
CyrilicFoneticALPHA.CyrilicPhonetic
btnKonwertuj_Click(object sender, EventArgs e)

52 {
53     StringBuilder sb_in = new StringBuilder(textBox1.Lines[1]); // utworzenie zmiennej klasy StringBuilder
54     string info_1 = textBox1.Lines[1]; // zmienna do dokonywania podglądu wartości w zakładce "lokalne"
55     int info_2 = textBox1.Lines[1].Length; // ilość znaków w 1-tej linii // zmienna do dokonywania podglądu wartości w zakładce "lokalne"
56     for (int j = 0; j < textBox1.Lines[1].Length; j++) // pętla odpowiadająca za przejście pomiędzy literami w 1-tej linii tekstu
57     {
58         pom1 = sb_in[j].ToString(); // konwersja na string
59         int info_3 = pom1.Length; // zmienna do dokonywania podglądu wartości w zakładce "lokalne"
60         if (Dziennik(pom1) == true) // warunek - jeżeli metoda dziennik zwraca odpowiedź true to...
61         {
62             // dopasowuje znak jeżeli spełniony jest warunek: dziennik == true
63             BIBLIOTEKA_ZNAKOW
64             if (Dziennik(pom1) == false) // warunek - jeżeli metoda dziennik zwraca odpowiedź false to...
65             {
66                 pom_sb += "[?]"; // staw w zmian za nierozpoznany znak: [?]
67                 NIE_cyry++;
68             }
69         }
70         calosc = calosc + pom_sb + nowalini; // połączenie w gotową do wydruku zmienną typu string
71         pom_sb = ""; // czyszczenie zmiennej pom_sb
72     }
73     textBox2.Text = Convert.ToString(calosc); // wydruk tekstu po dokonanych operacjach
74 }

75 private bool Dziennik(string info) // metoda "dziennik" sprawdzająca - czy dany znak znajduje się wśród znanych znaków
76 {
77     PRZYKIS USUWANIA TEKSTU
78     PRZYKIS RESETU
79     PRZYKIS KOPIONANIA TEKSTU z TB
80     // Zmienna "AutoScroll" służy do sprawdzenia czy jest włączona funkcja automatycznego przewijania
81     public static bool AutoScroll = false; // na początku autoprzewijanie jest WYŁĄCZONE
82     bool pom1 = false; // zmienna pomocnicza
83     public void btnAutoScroll_Click(object sender, EventArgs e) // metoda opisująca co ma się stać po kliknięciu przycisku "Funkcja autoprzewijania"
84     {
85         if (AutoScroll == false) // inicjacja działania przycisku bistabilnego
86         {
87             AutoScroll = true;
88             pom1 = true;
89         }
90         if (AutoScroll == true && pom1 == false)
91         {
92         }
93     }
94 }

```

Rysunek 2.3. – Fragment 2 skryptu z Visual Studio

```

Form1.cs* [Projekt]*
CyrilicFoneticALPHA.CyrilicPhonetic
btnKonwertuj_Click(object sender, EventArgs e)

691 bool pom1 = false; // zmienna pomocnicza
692
693 public void btnAutoScroll_Click(object sender, EventArgs e) // metoda opisująca co ma się stać po kliknięciu przycisku "Funkcja autoprzewijania"
694 {
695     if (AutoScroll == false) // inicjacja działania przycisku bistabilnego
696     {
697         AutoScroll = true;
698         pom1 = true;
699     }
700     if (AutoScroll == true && pom1 == false)
701     {
702         AutoScroll = false;
703     }
704     pom1 = false;
705 }
706
707 class SyncTextBox : TextBox // dziedziczenie klasy SyncTextBox po klasie TextBox // stworzenie zedyfikowanego textBox i dodanie go do toolbox (przybornika)
708 {
709     public SyncTextBox()
710     {
711         this.Multiline = true; // ustawienie wartości wskazującej
712         this.Scrollbars = Scrollbars.Vertical; // ustawienie które scrollbary powinny pojawić się w kontrolce wielowierszowej
713     }
714     public Control Wspolpracownik { get; set; } // wspolpracownik - do oznaczenia który textBox ma sterować którym, w trakcie działania funkcji autoprzewijania
715     private static bool scrolling; // zmienna pomocnicza typu bool
716     protected override void WndProc(ref Message m) // metoda do przetwarzania komunikatów Windows
717     {
718         base.WndProc(ref m);
719         if ((m.Msg == 0x115 || m.Msg == 0x100) && IsScrolling && Wspolpracownik != null && Wspolpracownik.IsHandleCreated && CyrilicPhonetic.AutoScroll == true) // warunek przy którym spełnieniu działa autoprzewijanie
720         {
721             scrolling = true;
722             SendMessage(Wspolpracownik.Handle, m.Msg, m.WParam, m.LParam); // parametry
723             scrolling = false;
724         }
725     }
726     [DllImport("user32.dll", CharSet = CharSet.Auto)] // wskazuje do przypisanego metody jest udostępniana przez niezarejestrowaną bibliotekę dołączoną (Dll) jako statyczny punkt wejścia
727     private static extern IntPtr SendMessage(IntPtr hWnd, int msg, IntPtr wp, IntPtr lp);
728 }
729
730
731
732
733
734
735

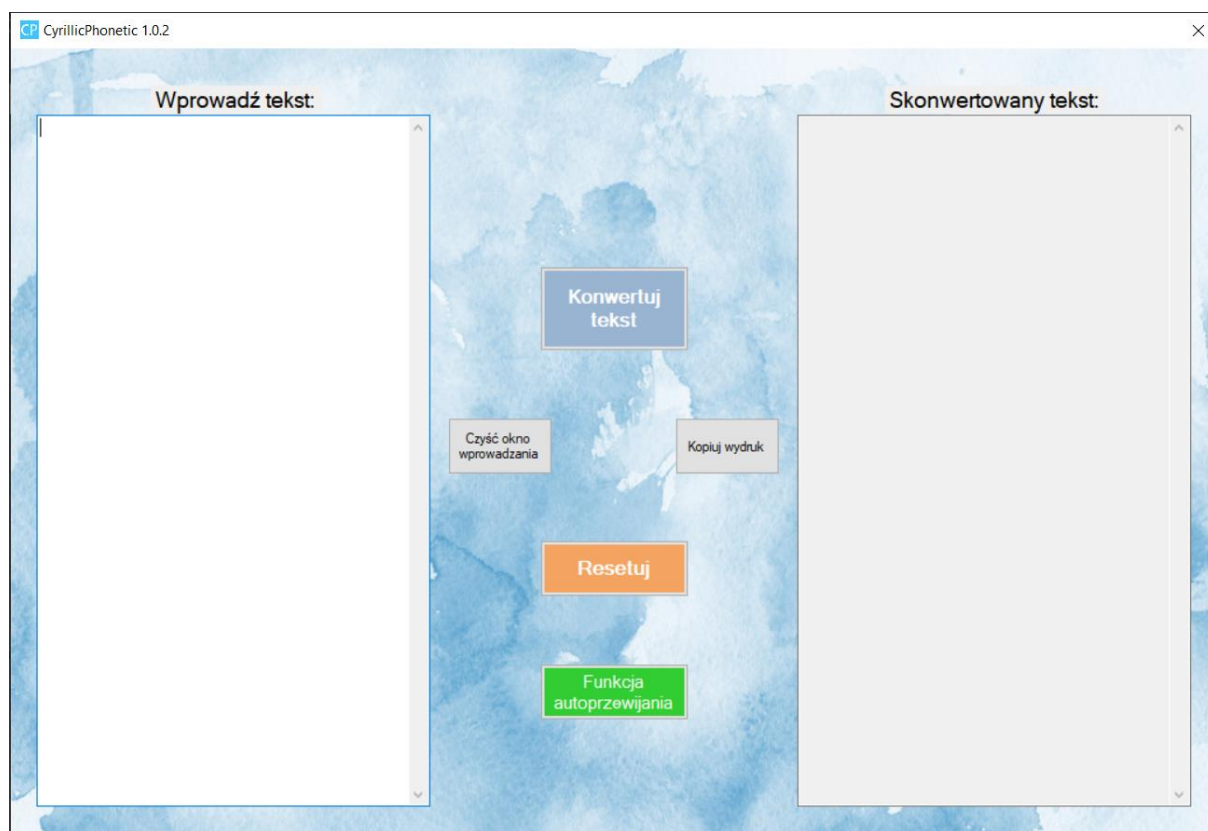
```

Rysunek 2.4. – Fragment 3 skryptu z Visual Studio

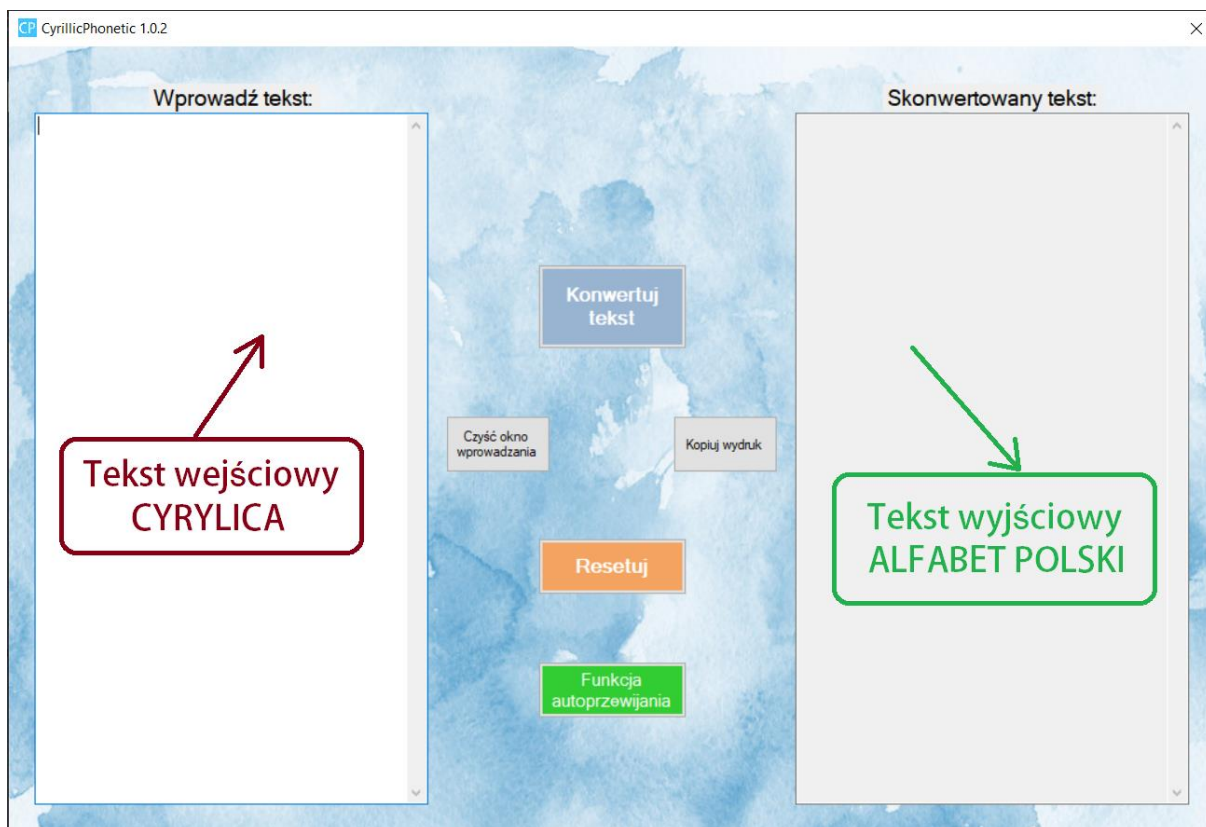
2.4. Front-end

Aplikacja posiada nadzwyczaj prosty interfejs graficzny. Na oknie wyświetlają się dwa okna tekstowe (textBox1 oraz textBox2). textBox1 służy do wprowadzania tekstu

wejściowego – tekstu zapisanego w cyrylicy. textBox2 natomiast jest elementem wyłącznie do odczytu i reprezentuje obszar zwrotu tekstu w formie znaków alfabetu polskiego. Rysunek poniżej:

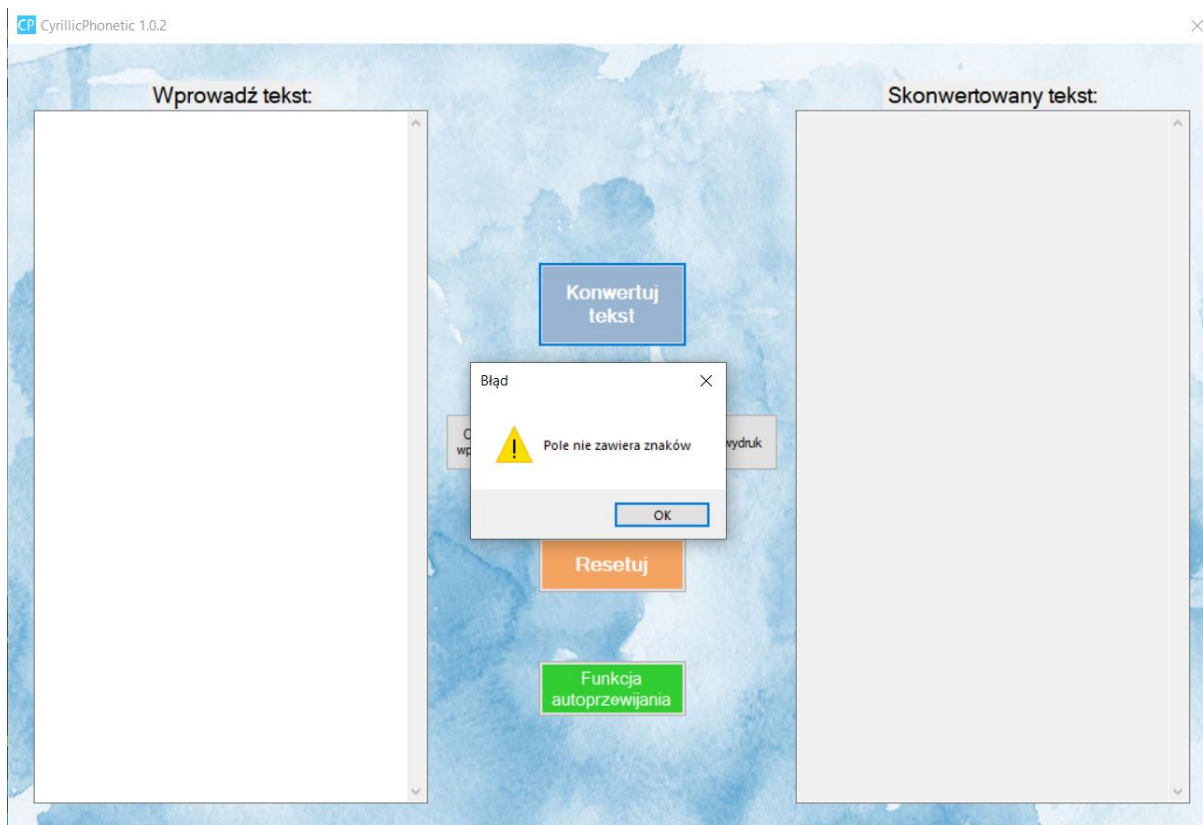


Rysunek 2.5. – Okno (menu aplikacji) interfejsu graficznego użytkownika



Rysunek 2.6. – Okno (menu aplikacji) interfejsu graficznego prezentowanego użytkownikowi

Na środku okna znajduje się **5 przycisków**. Pierwszy położony najwyżej, koloru niebieskiego, posiada etykietę – nazwę „**Konwertuj tekst**”. Uaktywnia on proces konwersji tekstu – głównego zadania aplikacji jeżeli znajduje się w nim niezerowa liczba znaków. W przeciwnym razie przy próbie naciśnięcia przycisku aktywuje się alert z informacją o braku tekstu w **textBox1**. Rysunek poniżej:

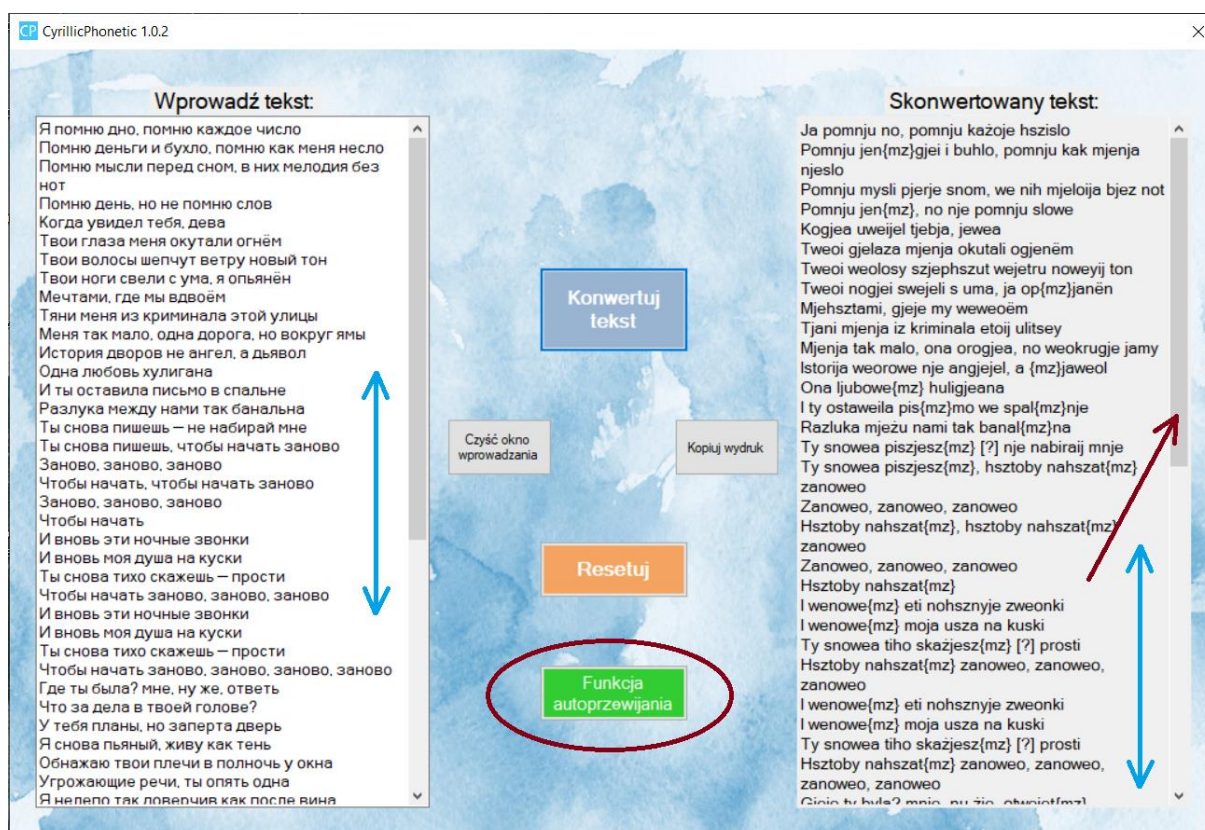


Rysunek 2.7. – alert z informacją o braku tekstu w textBox1

Po lewej stronie znajduje się przycisk z nazwą „**Czyść okno wprowadzania**”. Można za pośrednictwem niego usunąć tekst znajdujący się w textBox1 w lewym oknie. Po prawej zaś, znajduje się przycisk o nazwie „**Kopiuj wydruk**”, który umożliwia natychmiastowe skopiowanie całego fragmentu tekstu z przekonwertowanego do schowka.

Przycisk „**Resetuj**” umieszczony w centrum menu służy do usuwania tekstu z obu textBox-ów (lewego i prawego).

Przycisk położony najniżej, koloru zielonego, o nazwie „**Funkcja autoprzewijania**” **aktywuje funkcję auto-przewijania**, czyli pomocną funkcję w czytaniu i porównywaniu tekstów z obu textBox-ów. Działa ona na zasadzie takiej, że scrollbar umieszczony po prawej stronie textBox2, po aktywacji wspomnianej funkcji przewija zarówno tekst lewy jak i prawy textBox. Rysunek poniżej:



Rysunek 2.8. – Opis graficzny funkcji auto-przewijania

Program posiada jeszcze **alert**, który ujawnia się gdy w tekście wpisanym występują znaki obce (niebędące cyrylicą), w liczbie powyżej 10-ciu. Tekst alertu: „Konwertowany tekst zawiera znaki nienależące do Cyrylicy Znaki zostały oznaczone jako: [?]”.

3. Wnioski

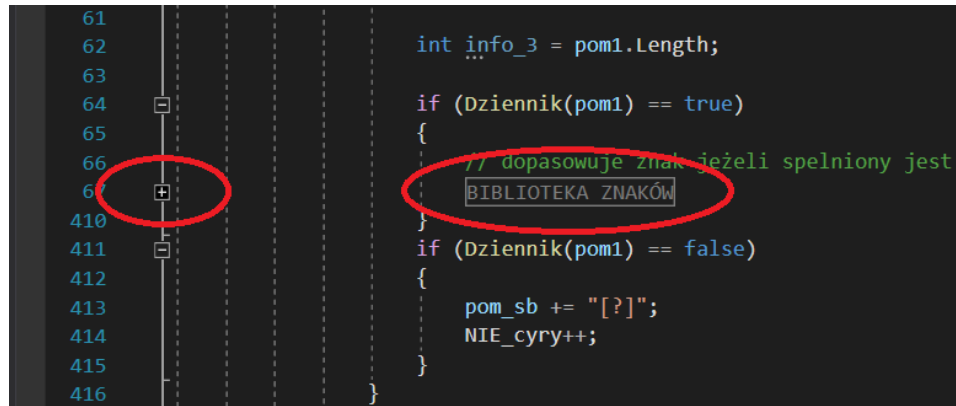
Aplikacja działa poprawnie. Osiągnięto zamierzony cel. Należy jednak zaznaczyć, że wydrukowany w textBox2 tekst (zwracany przez program) jest jedynie przybliżeniem brzmienia tekstu pisanego w cyrylicy i występować będą często większe lub mniejsze uchybienia od brzmienia rzeczywistego.

Należy również podkreślić, że w cyrylicy występują tzw. „Twardy znak” oraz „miękki znak”. Wyglądają one następująco (kolejno):

Ъ ъ	Ь ь
ь ь	Ь ь

Zmieniają one istotne brzmienie wyrazów przez co można napotkać większe różnice między brzmieniem rzeczywistym. Zostały one ujęte w sposób natępujący w tekście: twardy znak - {tz} i miękki znak - {mz}.

Część skryptu w MS Visual Studio została również zminimalizowana.



```
61  
62  
63  
64  
65  
66  
67  
410  
411  
412  
413  
414  
415  
416
```

```
int info_3 = pom1.Length;  
  
if (Dziennik(pom1) == true)  
{  
    // dopasowuje znak jeżeli spełniony jest  
    BIBLIOTEKA_ZNAKÓW  
}  
  
if (Dziennik(pom1) == false)  
{  
    pom_sb += "[?]";  
    NIE_cyry++;  
}  
}
```

The image shows a screenshot of the MS Visual Studio code editor. On the left, a vertical line of line numbers is visible, with lines 61 through 67 and 410 through 416 highlighted in blue. On the right, the corresponding C# code is displayed. Two red circles are drawn on the image: one around the line number 67 and another around the text 'BIBLIOTEKA_ZNAKÓW' in the code. The code itself is as follows: